数据流频繁模式挖掘算法设计*)

敖富江　颜跃进　黄　健　黄柯棣
(国防科技大学机电工程与自动化学院　长沙 410073)

摘　要　介绍了数据流频繁模式的概念和定义，提出了数据流频繁模式挖掘算法的通用数据流处理模型，详细总结了数据流频繁模式挖掘算法的三种分类方式："窗口模型"、"结果集类型"和"结果集精确性"。基于这些分类方法提出了数据流频繁模式挖掘算法的设计立方体，该立方体不仅涵盖了现有的数据流频繁模式挖掘算法，还对设计新的算法具有指导意义。基于设计立方体，分析了设计算法时应当采取的有效策略，旨在为设计新算法提供一个有力参考。最后讨论了数据流频繁模式挖掘的进一步研究工作。

关键词　数据流，频繁模式，设计立方体

Designing Algorithm on Mining Frequent Patterns in Data Streams

AO Fu-Jiang　YAN Yue-Jin　HUANG Jian　HUANG Ke-Di
(College of Mechanical Engineering and Automation, National University of Defense Technology, Changsha 410073)

Abstract　This paper introduces some concepts and definitions about frequent patterns in data streams, and presents the general data streams processing model for mining frequent patterns in data streams, and detailedly sums up three classifications of the algorithms on mining frequent patterns in data streams, including Window Model, the Type of Result Set and the Accuracy of Result Set. Based on these classifications, the cube for designing algorithm on mining frequent patterns in data streams is presented. The cube not only covers the exiting algorithms, but also allows for the design of new ones suited for various application requirements. Based on the cube, we analyse some valid strategies for designing the algorithm, aiming at presenting a powerful reference for designing new ones. Lastly, we discuss some future research works about mining frequent patterns in data streams.

Keywords　Data streams, Frequent pattern, Cube for designing algorithm

1　引言

数据流是近几年以来国内外研究的一个热点。它广泛地存在于多种领域，包括传感器网络、电信通话记录、气象监测与分析、股票分析、邮件过滤、网络监控与安全、Web 日志分析，以及大规模科学计算的数据分析(例如文[1]中研究的仿真数据流)等方面。对于它的概念，不同的文献给出了大致相同的定义[2~6]：数据流是一种连续、高速、无限、时变的有序序列。我们认为，只要是按照时间顺序先后到达的数据序列都可以被看作为数据流。具体到数据流研究领域，通常研究具有连续性、高速性、数据量无限性、数据内容随时间改变等特征的数据流。这些特征对于数据流的研究来说，更具有挑战性。连续性要求数据流处理算法只能是一遍(one-pass)算法；高速性要求数据流处理算法的实时性高；数据量无限性使得无法将全部数据存储到主存，甚至是永久存储器中；数据内容随时间改变通常会带来概念漂移问题，因此数据流处理算法必须具有处理概念漂移的能力。

对数据流的研究包括两个方面：数据流管理技术和数据流挖掘技术。数据流管理技术主要研究开发数据流管理系统所需要的一些技术，主要包括：数据流连续查询语言的设计、查询优化、资源管理(主要是内存)、近似查询操作、流系统的调度等。目前，已经出现了多种数据流管理系统原型，例如斯坦福大学的 STREAM 系统、布朗大学和麻省理工学院的 Aurora 系统、StatStream 系统、加州大学伯克里分校的 TelegraphCQ 系统、COUGAR 及乔治亚州工学院研制的 OpenCQ 系统、维斯康新迈迪讯实验室的 Niagara-CQ 系统等[7]。数据流挖掘技术主要研究数据流上的聚类、分类、频繁模式和时序分析等数据挖掘技术。

本文主要介绍数据流频繁模式挖掘技术。其中第 2 节介绍了一些基本定义，第 3 节概括了数据流频繁模式挖掘的数据流处理模型，第 4 节提出了"数据流频繁模式挖掘的设计立方体"，第 5 节基于设计立方体分析了设计算法时应当采取的策略，第 6 节介绍了进一步的研究工作，最后对全文进行了总结。

2　基本概念与定义

不同的数据流挖掘技术所挖掘的数据流类型不同。数据流频繁模式挖掘所处理的通常是事务(Transaction)性数据流，即按照时间顺序先后到达的事务。在分析了多篇文献的基础上[8,9,14~17]，我们给出了事务性数据流的如下形式化定义：

定义 1(事务性数据流)　令 $I=\{i_1,i_2,\cdots,i_m\}$ 为 m 个不

*)本文得到国家科学自然基金项目(项目编号：60573057)资助。**敖富江**　博士生，主要研究方向为数据仓库、数据挖掘和复杂大系统仿真；**颜跃进**　博士，主要研究方向为数据仓库和数据挖掘；**黄　健**　博士，主要研究方向为仿真标准、分布交互仿真技术和仿真通用软件；**黄柯棣**　教授，博士生导师，主要研究领域为仿真标准、仿真机软件与仿真算法、先进分布仿真技术和虚拟样机技术等。

同文字的集合，其中的文字称为项(Item)，则对于任意 $X \subseteq I$，称 X 为一个项集(如果 $|X|=k$，则称 X 为 k 项集)。事务 T 是一个项集，因此对于任意事务 T，存在 $T \subseteq I$。令 t 表示任一时间戳，T_t 表示在该时间戳到达的事务，则事务性数据流可以表示为 $\{\cdots, T_{t_{i-1}}, T_{t_i}, T_{t_{i+1}} \cdots\}$。

由于数据流的无限性，因此挖掘数据流上的频繁模式通常是指挖掘数据流上的某个窗口中的频繁模式。

定义 2(窗口内的频繁模式)　令 W 为数据流上的窗口，$f_P(W)$ 为窗口 W 某个模式 P 的频数，称 $S_P(W)=f_P(W)/|W|$ 为模式 P 在窗口 W 内的支持度，其中 $|W|$ 为窗口 W 的宽度。对于给定的窗口 W，如果 $S_P(W)$ 大于等于某个指定的支持度 $s(0<s<1)$，则称 P 是 W 内的频繁模式，简称 P 在 W 内频繁。

3　频繁模式挖掘算法的数据流处理模型

文[6,14]中分别给出了通用的数据流处理模型。在通用数据流处理模型的基础上，分析了一些已有的数据流频繁模式挖掘算法的处理方式后，我们概括了如下关于数据流频繁模式挖掘的数据流处理模型，如图 1 所示。

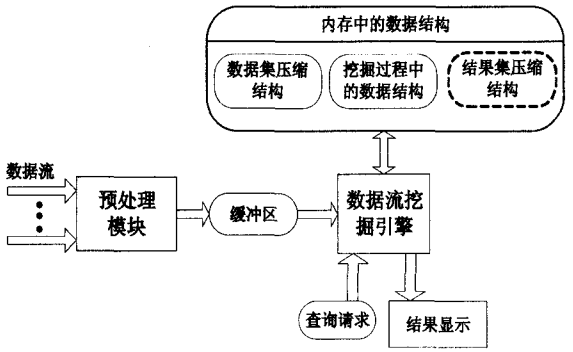

图 1　数据流频繁模式挖掘的数据流处理模型

该模型主要由三个模块组成：预处理模块、数据流挖掘引擎和结果显示模块。

对于到达的数据流，首先由预处理模块处理，并将处理过的事务数据放在临时缓冲区中。为了保证后续挖掘算法能够及时处理缓冲区中的数据，这期间可能会利用采样技术等进行近似处理。

数据流挖掘引擎负责处理查询请求和数据的挖掘。为了辅助数据挖掘和提高查询速度，算法通常需要在内存中维护一些数据结构。可以将这些数据结构分成三类：数据集压缩结构，用于挖掘过程的数据结构和结果集压缩结构。数据集压缩结构以紧缩的方式存放数据流中的事务数据，当数据到达缓冲区时，由数据流挖掘引擎将数据压缩到数据集压缩结构中；用于挖掘过程的数据结构是挖掘过程中需要应用到的一些数据结构；结果集压缩结构存放挖掘的结果。一般的数据流频繁模式挖掘算法都具有前面两种结构。另外，为了加速查询速度，有些算法还在内存中维护结果集压缩结构，当查询请求到达时，直接从结果集中获得相应的答案，但并不是每个算法都在内存中维护结果集。维护结果集的好处是能够加速查询速度，但不足之处是会牺牲算法的空间效率和时间效率。

结果显示模块以用户易理解的形式输出所挖掘的频繁模式。

另外，在挖掘时机上，存在两种方式：一种方式是，当每一条事务到达时，数据流挖掘引擎除了将其压缩到压缩数据集结构之外，并立即求出当前考察窗口中的结果集；另一种方式是，当事务到达时，只是将其压缩到压缩数据集结构中，只有当用户请求挖掘时，才对当前考察窗口中的事务数据进行挖掘，或者周期地对考察窗口中的数据进行挖掘。第一种方式的好处是能够立即了解结果的变化，文[15,16]中的算法采用了第一种方式。但当数据较稠密，结果集变化较频繁时，该方式的算法效率通常比第二种方式低。当用户请求获得结果的频率较低时，第二种方式比较适合。大多数算法采用第二种方式。

4　设计立方体

对于当前已经出现的数据流频繁模式挖掘算法，可以从三个典型的方面分别进行分类：窗口模型(Window Model)、结果集类型(the Type of Result Set)和结果集精确程度(the Accuracy of Result Set)。

由于数据流的连续性和无限性，对于数据流的挖掘来说，选择一个合适的窗口是非常重要的。文[9]中提出了三种数据处理模型：界标窗口模型(LW, Landmark Windows)、滑动窗口模型(SW, Sliding Windows)和衰减窗口模型(DW, Damped Windows)。根据这三种窗口模型，可以将当前的数据流频繁模式挖掘算法分为三类。

结果集类型包括三类：频繁项集(FI, Frequent Itemsets)、频繁闭项集(CFI, Closed Frequent Itemsets)、最大频繁项集(MFI, Maximal Frequent Itemsets)。对应地可以将挖掘算法分成三类。

数据流频繁模式挖掘算法通常可以分为精确算法(EA, Exact Algorithms)和近似算法(AA, Approximate Algorithms)。而近似算法又可以进一步分为两种类型：false positive 类型(FPAA)和 false negative 类型(FNAA)。前者的结果集中包含某些非频繁模式，而后者可能丢失某些频繁模式。因此，根据是否精确算法可以将挖掘算法分为三类：EA、FPAA、FNAA。

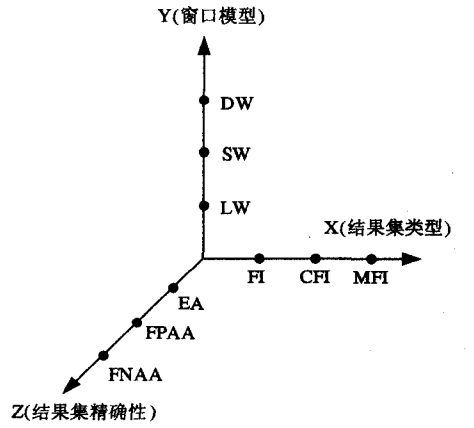

图 2　数据流频繁模式挖掘算法的设计立方体

由于数据流频繁模式挖掘算法的三种分类方式是相互独立的，因此可以将这三个方面分别作为空间中的一维(每一维是相互正交的)，构成数据流频繁模式挖掘算法的设计立方体。其中 X 轴为结果集类型(the Type of Result Set)，该轴由三个点组成：频繁项集(FI)、频繁闭项集(CFI)、最大频繁项集(MFI)；Y 轴为窗口模型(Window Model)，该轴由三个点组成：界标窗口模型(LW)、滑动窗口模型(SW)和衰减窗口模

型(DW);Z轴为是否精确算法(Exact or Approximate),该轴由三个点组成:精确算法(EA)、false positive类型近似算法(FPAA)和false negative类型近似算法(FNAA)。数据流频繁模式挖掘算法的设计立方体如图2所示。

对于当前存在的数据流频繁模式挖掘算法,可以根据设计立方体进行归类,将算法映射到设计立方体空间中的某一个点中,以研究算法的发展趋势。另外,针对特定的数据流应用,基于设计立方体,可以通过在每一维上选择最适合的方式构成最佳的频繁模式挖掘算法。因此,设计立方体对于数据流频繁模式挖掘算法的设计者来说,具有辅助设计最佳的挖掘算法的功能。

5 算法设计策略

面对一个特定应用,如何进行算法的设计呢?本节基于设计立方体,分别讨论了每一维中每种分类的适用范围,并分析了每种分类对算法的时间效率(time efficiency)和空间效率(space efficiency)的影响。

5.1 窗口模型

三种窗口模型之间不存在优劣问题,只是应用的范围不同。特定的应用适用于特定的窗框模型。

界标窗口模型是人们最早使用的一种窗口模型。采用界标窗口模型的算法挖掘从一个特定的时间点到当前时间的所有数据的频繁模式,该特定时间点称为"界标"。大量的早期算法基于这种模型,例如文[10,11,14,17~20,34]中的算法。它的优点是能够保存从某个起始点开始到当前的所有信息。但缺点也是显而易见的,首先无法在存储器中压缩所有的数据,因此通常的做法是将数据流分成多个桶(bucket),分别挖掘每一个桶的结果,然后在内存中维护近似的结果集,以回答用户的查询。但当人们仅对数据流的近期信息感兴趣时,这种模型不太适合。例如股票监测系统,对于最终用户来说,当前的实时信息更有意义。如果不和其他的数据更新技术组合,这种算法无法很好地处理连续、大容量的数据流数据。

滑动窗口模型适用于仅对当前数据感兴趣的应用。在滑动窗口模型中,算法仅查找并维护宽度为 w 的当前窗口中的频繁项集。例如文[15,21,22,33,34]中的算法处理了当前滑动窗口中的数据流。当数据项过期,超出了当前滑动窗口时,需要彻底消除它对当前挖掘结果的影响。对于滑动窗口模型,通常的处理方式是将窗口中的所有事务压缩到内存中特定的数据结构中;当事务到达时,将其压缩到数据结构中,当事务离开窗口时,将其从窗口中删除。它的另外一个优点是能够根据存储器和计算资源的情况动态维护窗口的大小,并且这种窗口模型适合于维护精确的结果集。

在衰减窗口模型中,每一个事务具有一个权重,当事务"衰老"时,它的权重随之降低。事务越老,它对项集的频率的贡献越小。文[23,24]中的算法使用了这种数据处理模型。当不同时期的事务对结果的贡献不同时,可以采用这种窗口模型。衰减窗口模型通常与界标窗口模型或滑动窗口模型组合起来使用。

5.2 结果集类型

挖掘数据流的频繁项集是最早被研究的内容。文[8,10,24,25~27,34~36]中的算法挖掘了数据流的频繁项集。由于频繁项集的子集也是频繁项集,因此,挖掘频繁项集的一个不足之处在于可能产生组合爆炸问题,严重影响算法的时空效率[30,31]。现在的研究趋势是挖掘数据流上的频繁闭项集(例如文[15]中的算法)和最大频繁项集(例如文[14,16,28]中的算法)。

最大频繁项集中包含了所有的频繁项集,它的数目远小于频繁项集,并且很多应用也只需要挖掘最大频繁项集,因此对于存在大量强模式、长模式和要求阈值较低的应用,可以考虑挖掘最大频繁项集。

但最大频繁项集的一个不足之处在于,它不包含子项集的支持度,因此会丢失信息。而频繁闭项集则不存在这种问题。一个频繁项集 I 必定是一个或多个频繁闭项集的子集,并且 I 的支持度等于这些频繁闭项集中支持度最大的项集的支持度[32]。频繁闭项集的大小介于频繁项集和最大频繁项集之间。在数据量方面,三者具有如下关系:

$$FI>CFI>MFI \tag{1}$$

最大频繁项集和频繁闭项集的结果集都远小于频繁项集,因此对于空间效率要求较高的数据流挖掘算法来说,是比较好的选择。当不需要了解所有子项集频率时,可以考虑挖掘最大频繁项集,否则可以选择挖掘频繁闭项集。

另外,有些应用可能只对满足一定约束条件的频繁模式感兴趣。因此,如果频繁模式挖掘算法只挖掘满足用户指定约束条件的频繁模式,则将大量节省用户处理不感兴趣的频繁模式的时间[29],因此可以考虑挖掘数据流上的基于约束的频繁模式。

5.3 结果集精确性

通常,数据流上的频繁模式挖掘算法对空间效率和时间效率要求较高,而近似算法的时空效率比精确算法的高。因此大多数算法采用近似算法,仅有少量算法采用精确算法。

精确算法需要额外的开销,通常的做法是在内存中维护所有的事务,或者维护所有项集的频率。无论是采用哪一种方法都会极大地降低空间效率,因此仅适用于特定的窗口模型,或挖掘特定类型的数据流。例如文[15]中的算法仅挖掘滑动窗口中的精确频繁闭项集,文[12]中的算法仅挖掘精确的短频繁项集。

而近似算法需要保存的信息较少。对于近似算法,是选择false positive类型的近似算法,还是选择false negative类型的近似算法呢?当前大多数算法选择false positive近似类型。它的优点是算法简单,很容易保证算法是no false negative类型(即算法不丢失频繁模式)。

false negative类型的近似算法的缺点是会漏掉某些频繁模式,但它的空间效率要比false positive类型的近似算法高。为了解决概念漂移问题(即某些非频繁项集在将来可能变为频繁项集),近似算法通常在存储器中维护估测频率高于某个较小的阈值 ε 的所有项集,不管它们实际上是频繁的还是非频繁的。为了保证不漏掉频繁项集,false positive类型的近似算法相对于false negative类型的近似算法来说,需要维护更多的非频繁项集。因此,对于空间效率要求非常高的算法,可以考虑采用false negative类型的近似算法,即通过漏掉一些频繁项集而换取空间效率。

5.4 时空效率分析

窗口模型对于时空效率的影响比较直观。显然,窗口中所含的数据量越多,算法的时空效率越低。图3中给出了界标窗口模型和滑动窗口模型对时空效率的影响趋势(衰减窗口模型通常与这两种窗口模型组合使用,因此没有被考虑)。

对于结果集来说,当结果集中所包含的结果数目越多时,算法的时空效率越低。因此在时空效率方面,挖掘全频繁项

集的时空效率最低，其次是挖掘频繁闭项集，最好的是挖掘最大频繁项集。但完全频繁项集和频繁闭项集中包含了完全的支持度信息，而最大频繁项集的信息不完全。

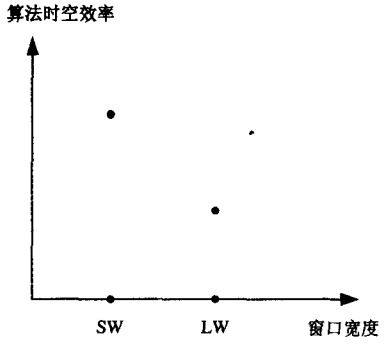

图 3　不同窗口模型下算法的时空效率

对于精确算法和近似算法来说，精确算法的时空效率不如近似算法，而 false negative 类型的近似算法的时空效率又比 false positive 类型的近似算法高。图 4 中给出了不同结果集精确性和结果集类型对时空效率的影响。

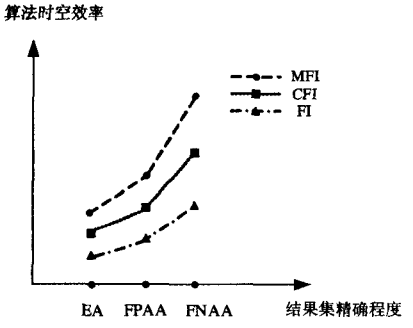

图 4　不同结果集精确性和结果集类型下算法的时空效率

以上时空效率分析并未考虑算法的具体设计和实现。在不同的设计和实现方案下，一个蹩脚的挖掘最大频繁项集的算法的时空效率有可能比挖掘频繁项集的算法低。但在相同的设计和实现方案下，通常具有以上相对关系。

通过以上分析可以看出，当对算法的时空效率要求特别高时，可以考虑设计挖掘滑动窗口中最大频繁项集的 false negative 类型的近似算法（即 SW-MFI-FNAA 类型的算法）。近两年来，挖掘数据流上的频繁闭项集和最大频繁项集得到了不少学者的研究[14~16,28]。

6　进一步的工作

当前，虽然已经出现了一些比较经典的数据流频繁模式挖掘算法，但在这个领域仍然存在一些方面需要进一步改进和提高。设计这类算法的目标是尽可能获得更高的空间效率和时间效率，对于近似算法，还要尽可能获得更好的精度。尤其是随着移动设备的普及，利用移动设备进行数据挖掘使得对算法的空间效率和时间效率的要求更苛刻。我们认为，可以在以下几个方面开展进一步的研究。

(1)近似算法的研究

衡量近似算法的两个重要指标是 recall 和 precision。令 A 为真频繁项集，B 为近似算法所获得的频繁项集，则

$$recall=\frac{|A\cap B|}{|A|} \tag{2}$$

$$precision=\frac{|A\cap B|}{|B|} \tag{3}$$

即 *recall* 为所挖掘的真频繁项集数目与完全的真频繁项集数目之比，*precision* 为所挖掘的真频繁项集数目与所挖掘的所有频繁项集数目之比。显然 *recall* 和 *precision* 的值都小于或等于 1，并且对于好的近似算法来说，*recall* 和 *precision* 的值应当尽可能地接近于 1。

当前的 false positive 算法通常保证为 no false negative[14,24,26]，而 false negative 算法通常保证为 no false positive[13]。前者的 *recall* 值等于 1，*precision* 值小于 1；后者的 *recall* 值小于 1，*precision* 值等于 1。虽然能够做到其中一个值等于 1，但通常另外一个指标并不是很好。因此可以考虑研究两个指标均佳的近似算法。

(2)数据集和结果项集的压缩数据结构研究

压缩数据结构对算法的空间效率和时间效率都会产生影响。好的数据集压缩数据结构不但应当能够提供高压缩率，而且对后续的挖掘过程也应当提供高效支持。而在设计结果集压缩数据结构时，不但要考虑它的压缩率，也要考虑它对查询速度的影响。

(3)高效挖掘过程的研究

针对特定的压缩数据集，挖掘过程通常需要利用一些临时数据结构进行挖掘，这些临时数据结构将对算法的空间效率产生重要影响。因此必须综合考虑压缩数据集结构和挖掘过程，使得算法的总空间效率和时间效率高。

(4)数据流预处理技术研究

预处理过程对后续的挖掘过程具有重要的影响。在算法的执行过程中，所能够使用的内存资源和计算资源是有限的，并且可能是动态变化的，因此可以考虑设计自适应的预处理过程。

(5)与移动 Agent 技术的结合

移动 Agent 是一种可以在网络中自主移动的代码。利用移动 Agent 技术能够将时间和空间耗费高的计算自动转移到网络其他机器中，因此对于利用普氏设备进行数据挖掘来说具有重要意义。

总结　数据流频繁模式挖掘是数据流挖掘研究的一个重要内容，目前已经出现了一些非常经典的算法。本文按照三个方面对这些算法进行了分类总结，设计了数据流频繁模式挖掘算法的设计立方体，并基于设计立方体讨论了设计频繁模式挖掘算法时所应采取的策略，另外对进一步的研究内容进行了探讨。

衡量数据流上频繁模式挖掘算法优劣的两个重要指标是算法的空间效率和时间效率，对于近似算法，还要考虑算法的精度问题。但这三者往往互相矛盾，改进某一个方面，很可能会降低另外两个方面，因此设计算法时需要综合考虑这三个指标。针对具体的应用环境，必要时可以考虑进行一些折中，即牺牲某一方面，以获取总体性能的提升，或者牺牲某两个方面，以改进某一个方面。

参 考 文 献

1　Abdulla G, Critchlow T, Arrighi W. Simulation Data as Data Streams. ACM SIGMOD Record, 2004, 33(1): 89～94

2　Golab L, Özsu M T. Issues in Data Stream Management. ACM SIGMOD Record, 2003, 32(2): 5～14

3　Gaber M M, Zaslavsky A, Krishnaswamy S. Mining data streams: A Review. ACM SIGMOD Record, 2005, 34(2): 18～

26
4 Babcock B, Babu S, Datar M, et al. Models and issues in data streams. In: Popa L, ed. Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symp on Principles of Database Systems. Madison: ACM Press, 2002. 1～16
5 Nan Jiang, Le Gruenwald. Research Issues in Data Stream Association Rule Mining. ACM SIGMOD Record, 2006, 35(1): 14～19
6 潘云鹤,王金龙,徐从富. 数据流频繁模式挖掘研究进展. 自动化学报,2006,32(4):594～602
7 刘学军,徐宏炳,董逸生,等. 数据流管理技术. 计算机科学,2005,32(4):6～10
8 张昕,李晓光,王大玲,等. 数据流中一种快速启发式频繁模式挖掘方法. 软件学报,2005,16(12):2099～2105
9 Zhu Y, Shasha D. StatStream: Statistical monitoring of thousands of data streams in real time. In: Bernstein P, Ioannidis Y, Ramakrishnan R, eds. Proc. of the 28th Int'l Conf on Very Large Data Bases, Hong Kong: Morgan Kaufmann, 2002. 358～369
10 Li H, Lee S, Shan M. An efficient algorithm for mining frequent itemsets over the entire history of data streams. In: Proceedings of the first International Workshop on Knowledge Discovery in Data Streams, held in conjunction with the 15th European Conference on Machine Learning (ECML 2004) and the 8th European Conference on the Principles and Practice of Knowledge Discovery in Databases (PKDD 2004). Pisa, Italy, 2004
11 Manku G, Motwani R. Approximate frequency counts over data streams. In: Proc. of the Twenty-eighth International Conference on Very Large Data Bases. Hong Kong, China, 2002. 346～357
12 Li Yang, Sanver M. Mining Short Association Rules with One Database Scan. In: Arabnia R, ed. Proc. of the International Conference on Information and Knowledge Engineering. Las Vegas, Nevada, USA: CSREA Press, 2004. 392～398
13 Yu J Xu, Chong Zhihong, Lu Hongjun, et al. False positive or false negative: Mining frequent itemsets from high speed transactional data streams. In: Proc. 30th Intl Conf Very Large Data Bases. San Francisco: Morgan Kaufmann, 2004. 204～215
14 Li H, Lee S, Shan M. Online mining (recently) maximal frequent itemsets over data streams. In: Proc. of the Fifteenth International Workshops on Research Issues in Data Engineering: Stream Data Mining and Applications. Tokyo, Japan: IEEE Press, 2005. 11～18
15 Chi Y, Wang H, Yu P S, et al. Moment: maintaining closed frequent itemsets over a stream sliding window. In: Proc. of the Fourth IEEE International Conference on Data Mining, Brighton, UK: IEEE Press, 2004. 59～66
16 Mao Guojun, Wu Xindong, Liu Chunnian. Online Mining of Maximal Frequent Itemsequences from Data Streams: [Technical Report]. CS-05-07, University of Vermont, 2005
17 Charikar M, Chen K, Farach-Colton M. Finding frequent items in data streams. In: Proc. of the International Colloquium on Automata, Languages and Programming (ICALP). Malaga, Spain: Springer, 2002. 693～703
18 Demaine E, L'opez-Ortiz A, Munro J I. Frequency estimation of Internet packet streams with limited space. In: Mohring R H, Raman R, eds. Algorithms—ESA 2002, Proc of the 10th Annual European Symp, Rome: Springer-Verlag, 2002. 348～360
19 Jin C, Qian W, Sha C, et al. Dynamically maintaining frequent items over a data stream. In: Carbonell J, ed. Proc. of the 2003 ACM CIKM Intl Conf on Information and Knowledge Management. New Orleans: ACM Press, 2003. 287～294
20 Karp R M, Shenker S, Papadimitriou C H. A simple algorithm for finding frequent elements in streams and bags. ACM Transactions on Database Systems, 2003, 28(1): 51～55
21 Chang Joong Hyuk, Lee Won Suk. A Sliding Window Method for Finding Recently Frequent Itemsets over Online Data Streams. Journal of Information Science and Engineering, 2004, 20(4): 753～762
22 Lin C H, Chui D Y, Wu Y H, et al. Mining Frequent Itemsets from Data Streams with a Time-Sensitive Sliding Window. In: Proceedings of the Fifth SIAM International on Data Mining. Newport Beach, USA, 2005
23 Chang J, Lee W S. Finding recent frequent itemsets adaptively over online data streams. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington, USA: ACM Press, 2003. 487～492
24 Giannella C, Han Jiawei, Pei Jian, et al. Mining Frequent Patterns in Data Streams at Multiple Time Granularities. Data Mining: Next Generation Challenges and Future Directions, AAAI/MIT, 2003. 191～212
25 刘学军,徐宏炳,董逸生,等. 挖掘数据流中的频繁模式. 计算机研究与发展, 2005,42(12): 2192～2198
26 Manku G S, Motwani R. Approximate frequency counts over data streams. In: Bernstein P, Ioannidis Y, Ramakrishnan R, eds. Proc. of the 28th Int'l Conf. on Very Large Data Bases, Hong Kong: Morgan Kaufmann, 2002. 346～357
27 Cheng J, Ke Yiping, Ng W. Maintaining Frequent Itemsets over High-speed Data Streams. In: Proc. of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2006), Singapore, 2006
28 Lee D, Lee W. Finding maximal frequent itemsets over online data streams adaptively. In: Proceedings of the Fifth IEEE International Conference on Data Mining. Houston, USA: IEEE Press, 2005. 266～273
29 ZakiM J, Psrthasarathy S, Ogihara M, et al. New Algorithms for Fast Discovery of Association Rules. In: Proc. of the 3rd Int'l Conf on Knowledge discovery in Databases (KDD'9 7), 1997. 293～286
30 Grahne G, Zhu Jianfei. Efficiently Using Prefix-trees in Mining Frequent Itemsets. In: Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Melbourne, Florida, USA, 2003
31 颜跃进,李舟军,陈火旺. 基于 FP-Tree 有效挖掘最大频繁项集. 软件学报,2005,16(2):215～222
32 Wang Jianyong, Han Jiawei, Pei Jian. CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. In: Proc. of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA: ACM Press, 2003. 236～245
33 Arasu A, Manku G S. Approximate counts and quantiles over sliding windows. In: Proc. of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Paris, France: ACM Press, 2004. 286～296
34 Chang J, Lee W S. estWin: adaptively monitoring the recent change of frequent itemsets over online data streams. In: Proc. of the Twelfth International Conference on Information and Knowledge Management, New Orleans, USA: ACM Press, 2003. 536～539
35 Giannella C, Han J, Robertson E, et al. Mining frequent items. ets over arbitrary time intervals in data streams: [Technical Report]. tr587, Indiana University, 2003
36 Jia Lifeng, Zhou Chunguang, Wang Zhe, et al. SuffixMiner: Efficiently Mining Frequent Itemsets in Data Streams by Suffix-Forest. In: Proc. of Fuzzy Systems and Knowledge Discovery, Changsha, China, 2005. 592～595
37 Suffix-Forest. In: Proc. of Fuzzy Systems and Knowledge Discovery, Changsha, China, 2005. 592～595